PATENT APPLICATION COVER LETTER ATTORNEY DOCKET: 2207/5482

ASSISTANT COMMISSIONER FOR PATENTS
WASHINGTON, D.C. 20231

TITLE OF INVENTION:

SYSTEM AND METHOD FOR EVENT AND MICRO-BRANCH MISPREDICTION RECOVERY

INVENTORS:

Reynold V. D'Sa
Robert F. Krick
Rebecca E. Hebda
Alan B. Kyker

EVENT AND MICRO-BRANCH MISPREDICTION RECOVERY

Field Of The Invention

The present invention is directed to improvements to an instruction pipeline in a microprocessor. In particular, the present invention is directed to a system and method for event and micro-branch misprediction recovery in an instruction pipeline.

Background Information

Modern microprocessors include instruction pipelines in order to increase program execution speeds. Instruction pipelines typically include a number of units, each unit operating in cooperation with other units in the pipeline. One exemplary pipeline, found in, for example, Intel's Pentium® Pro microprocessor, includes an instruction fetch unit (IFU), an instruction decode unit (ID), a micro-code sequencer (MS), an allocation unit (ALLOC), an instruction execution unit (EX) and a write back unit (WB). The instruction fetch unit fetches program instructions. The instruction decode unit decodes macro-code instructions into a set number of micro-ops. However, if the macro-code instruction decodes into an a number of micro-ops that is greater than the set number, control is passed to the micro-code sequencer. The micro-code sequencer then provides the remaining micro-ops. The micro-code sequencer is also responsible for providing instructions to the execution unit when the processor must execute micro-code, for example, during event recovery. The allocation unit assigns a sequence number to each micro-op and stores each micro-op in an instruction pool. The execution unit executes the micro-ops. Finally, the write back unit retires instructions.

During operation, the execution unit executes the micro-ops in the instruction pool in any order possible as data and execution units required for each micro-op becomes available. If the processor "events" while retiring an instruction, that is, something occurs in the microprocessor that requires the microprocessor to discontinue processing the instructions from the current instruction path, and instead execute specific micro-code instructions, the microprocessor must have a fast way to recover the proper machine state.

Accordingly, for each micro-op it processes, the allocation unit stores in a branch information table (BIT) a pointer to the next linear macro-code instruction (NLIP). Then, when the microprocessor events, the retirement logic determines the appropriate

instruction address with which to restart the instruction pipeline, e.g., the instruction pointed to by the NLIP or another address.

In the Intel Pentium® Pro microprocessor, if the micro-code sequencer determines that the instruction pipeline should be restarted at the current instruction, the address of the current instruction must be calculated, i.e., the address of the current instruction is NLIP minus the length of the current instruction. Since macro-code instructions are not a uniform length, the instruction length of each instruction is passed in a data path, along with the instruction itself. Moreover, in the Intel Pentium® Pro microprocessor, other information needed by the micro-code sequencer, such as, for example, an indication as to whether a particular micro-op originated from the micro-code sequencer or another pipeline unit, is also transmitted in a dedicated data path. Data paths in microprocessors are an expensive resource. Accordingly, there is a need to reduce the number of data paths associated with the instruction pipeline.

Summary Of The Invention

In accordance with an exemplary embodiment of the present invention, a system for storing information in an instruction pipeline is provided, the instruction pipeline including a number of pipeline units. One pipeline unit decodes a macro-code instruction into a number of micro-ops. A second pipeline unit assigns a first identifier to each of the micro-ops. The system includes a register for storing information related to the macro-code instruction. The system also includes a table having a number of entries. Each of the entries in the table receives the stored information from the first register. Additionally, each of the table entries corresponds to a one of the micro-ops. The table entries are indexed by the identifier assigned to corresponding micro-op.

Brief Description Of The Drawings

Fig. 1 is a block diagram of an exemplary instruction pipeline in accordance with the present invention.

Fig. 2 illustrates the format of an exemplary trace branch information table.

Fig. 3 is a flow diagram of the exemplary embodiment of the present invention.

Fig. 4 is a flowchart of event recovery processing in accordance with the exemplary embodiment of the present invention.

Fig. 5 is a flowchart of micro-branch misprediction recovery processing in accordance with the exemplary embodiment of the present invention.

Detailed Description

Instruction Pipeline Overview: Referring now to the drawings, and initially to Fig. 1, there is illustrated an exemplary embodiment of the present invention. This embodiment illustrates the present invention as applied to, for example, an instruction pipeline for processing Intel Architecture (i.e., x86) instructions (e.g., IA-32). As a person of ordinary skill in the art will understand, however, the present invention may be applied to instruction pipelines of other processor architectures, such as, for example, RISC and CISC architectures, or any processor architecture that includes the use of an instruction pipeline.

As illustrated in Fig. 1, in the exemplary embodiment of the present invention, the instruction pipeline **100** includes seven major stages or units, although each of the pipeline stages or units may actually be comprised of a number sub-stages. As illustrated,the instruction pipeline **100** includes an instruction fetch unit (IFU) **110**, an instruction decode unit (ID) **120**, a trace cache unit(TC) **130**, a micro-code sequencer (MS) **135**, an allocation unit (ALLOC) **140**, an execution unit (EX) **150**, and a write-back unit (WB) **160**. It will be understood, however, that the number of pipeline units in the pipeline **100** (and, the function of each unit, for that matter) may be different than that described in connection with the exemplary embodiment, depending on, for example, the microprocessor architecture.

The instruction fetch unit **110** fetches, for example, program macro-code instructions from memory **111** (e.g., main memory, cache memory, or any other storage or memory device) and pushes the fetched instructions into the pipeline 100 to the next downstream pipeline unit. Although in this exemplary embodiment, macro-code instructions are Intel Architecture instructions, other instruction types such as, for example, RISC or CISC instructions, or any other type of instruction may instead by fetched.

During operation, the instruction fetch unit **110** fetches instructions from memory **111** in order to provide the pipeline **100** with a stream of instructions. If a fetched instruction is

a branch instruction, the instruction fetch unit **110** must determine whether to next fetch the instruction at the next sequential address, or the instruction at the branch target address. Accordingly, the instruction fetch unit **110** uses branch prediction circuitry (BTB) **113,** to predict whether or not a branch instruction will be taken or not taken. If the branch is predicted as "taken," the instruction fetch unit **110** fetches the instruction at the branch target address. If the branch is predicted as "not taken," the instruction fetch unit **110** fetches the next sequential instruction. In any event, the instruction fetch unit **110** pushes the fetched macro-code instruction to the next downstream pipeline unit.

The instruction decode unit **120** receives the macro-code instruction from the instruction fetch unit **110** in first-in, first-out (FIFO) order. The instruction decode unit **120** then decodes the macro-code instructions into, for example, fixed-length RISC instructions called micro-ops or uops. Each macro-code instruction may decode to one or a number of micro-ops. Each of these micro-ops is assigned an identifier, e.g., a sequence number, by the allocation unit **140**, and each is temporarily stored in an instruction pool **141**. Of course, as will be understood by a person of ordinary skill in the art, in some microprocessor architectures, instructions do not require decoding. In a pipeline in such a system, therefore, an instruction decoder (i.e., instruction decode unit **120**), for example, would not be needed.

If instruction decode unit **120** receives a macro-code instruction from instruction fetch unit **110** that is complex, for example, a macro-code instruction that decodes to more than four micro-ops, instruction decode unit **120** provides only the first four micro-ops. Micro-code sequencer **135** then provides the remaining micro-ops. (Also, as illustrated in Fig. 1, micro-ops provided to instruction pool **141** by instruction decode unit **130** pass through micro-code sequencer **135**.)

Micro-code sequencer **135** transforms all complex macro-code instructions into a corresponding set of micro-ops (the corresponding micro-ops are retrieved, for example, from a ROM **135a** illustrated in Fig. 3). In certain cases, if the decoded macro-code instruction includes a micro-branch instruction (a micro-op that is of branch type), micro-code sequencer **135** predicts branch direction, i.e., whether the branch will be taken or not taken. In the exemplary embodiment, micro-code sequencer **135** may make "static" branch predictions. In particular, micro-code sequencer **135** may always predict, for example, that an unconditional micro-branch will be taken, an conditional backward micro-branch will be taken, and a conditional forward micro-branch will not be taken.

Micro-code sequencer **135** then transmits to allocation unit **140** only those micro-ops along the predicted path.

In the exemplary embodiment of the present invention, instruction pipeline **100** includes an additional source of program instructions. In particular, trace cache unit **130** stores instruction sequences in the form of micro-ops (i.e., instruction traces) in high speed cache memory in order to later provide these instructions to allocation unit **140** for execution by execution unit **150**. The structure and operation of decoded cache unit **130** is described in further detail in U.S. Patent No. 5,381,533 to Peleg et al.

Decoded cache unit **130** controls whether the source for instructions entering instruction pool **141** is instruction fetch unit **110** (via instruction decode unit **120**) or decoded cache unit **130**. In particular, decoded cache unit **130** continuously snoops the instruction path **136** between micro-code sequencer **135** and allocation unit **140**. If decoded cache unit **130** recognizes that a particular instruction detected along the snooped instruction path **132** corresponds to a "trace head" (i.e., the first instruction in an instruction trace) are stored at decoded cache unit **130** (i.e., a decoded cache hit), decoded cache unit **130** signals instruction fetch unit **110** to discontinue fetching instructions. Instead, decoded cache unit **130** provides the appropriate instructions to allocation unit **140** from its cache memory. When decoded cache unit **130** detects that further necessary instructions are not in cache (i.e., a decoded cache miss), decoded cache unit **130** instructs instruction fetch unit **110** to recommence fetching instructions at an address provided by decoded cache unit **130.** Decoded cache unit **130** then discontinues providing instructions to allocation unit **140**.

In the exemplary embodiment of the present invention, execution unit **150** obtains instructions from the instruction pool **141**. Execution unit **150** executes the micro-ops in the instruction pool **141** in any order possible as data and execution units required for each micro-op becomes available. Accordingly, execution unit **150** is an out-of-order (OOO) portion of the pipeline. In other microprocessor architectures, pipeline **100** could include, for example, an execution unit that processes instructions in-order, or in some predetermined order.

Finally, write back unit **160** "retires" each executed micro-op. That is, write back unit **160** commits the result of each micro-op execution to the processor's register set in the order of original program flow. Thus, write back unit **160** is an in-order rear end of the

pipeline. Of course, in a microprocessor architecture in which instructions are executed in an in-order sequence, the instructions may not need to be "retired," thus, pipeline **100** may not include a write back unit.

In accordance with the present invention, certain information regarding each micro-op processed in instruction pipeline **100** is stored in a table. This information may be later used by machine micro-code, for example, during event recovery or during micro-branch misprediction recovery. In accordance with the present invention, a trace branch information table (TBIT) **142** stores information for each micro-op processed. Fig. 2 illustrates the format of an exemplary TBIT **142**.

TBIT Format: Referring now to Fig. 2, each entry **210** in TBIT **142** includes i) a sequence number field **215**; ii) an NLIP field **220**; iii) a BLIP field **225**; iv) a branch prediction field **230**; v) an IPdelta field **235**; vi) a uip field **240**; and vii) an MSIssue field **245**. Each entry **210** corresponds to, for example, one micro-op.

As described above, allocation unit **140** assigns a sequence number to each micro-op. After each micro-op is assigned a sequence number, TBIT **142** stores in sequence number field **215** the sequence number assigned to the current micro-op. Since the sequence number uniquely identifies each micro-op, the sequence number field **215** may be used for indexing TBIT **142**. In an alternate embodiment, the sequence number is not stored in TBIT **142**, and is simply used as an index into the table.

In NLIP field **220,** the address of the next linear macro-code instruction (NLIP), relative to the macro-code instruction associated with the current micro-op is stored.

If the macro-code instruction associated with the current micro-op is a branch instruction, the branch target address (BLIP) is stored in BLIP field **225**. Otherwise this field is filled in with "don't cares" (e.g., zeroes).

Additionally, if the macro-code instruction associated with the current micro-op is a branch instruction, a branch prediction bit is stored in branch prediction field **230**. For example, the branch prediction bit indicates whether the branch was predicted as upstream prediction circuitry as taken or not taken ("1" or "0", respectively).

The length of the macro-code instruction associated with the current micro-op is stored in IPdelta field **235**. Thus, the address of the macro-code instruction associated with the current micro-op may be determined by subtracting this length from the value stored in NLIP field **220**, i.e., current macro-code instruction address = NLIP - IPdelta. In an alternative embodiment, a current linear address pointer (CLIP) may be stored instead of the NLIP. In that case, the next linear address (i.e., the NLIP) may be calculated by adding the IPdelta to the CLIP.

In the exemplary embodiment of the present invention, the MSissue field **245** is a single bit field. If the micro-op originated from micro-code sequencer **135**, the bit is turned on (i.e., "1"). Otherwise, the bit is turned off (i.e., "0"), indicating that the micro-op originated either from instruction decode unit **120** or decoded cache unit **130**.

If the bit in MSissue field **245** is turned on, a pointer to the current micro-op is stored in the uip field **240**.

TBIT Maintenance: Fig. 3 illustrates in detail portions of instruction pipeline **100** pertinent to the maintenance and use of TBIT **142**. As illustrated, in accordance with the exemplary embodiment of the present invention, a recirculation register **310** is provided. Recirculation register **310** is coupled to path **136** (between micro-code sequencer **135** and allocation unit **140**). Recirculation register **310** is also coupled to TBIT **142** through a multiplexer **315**. Allocation unit **140** is coupled to TBIT **142** through multiplexer **315**.

In operation, information that is common to all of the micro-ops associated with a particular macro-code instruction is stored in recirculation register **310**. In particular, for each decoded macro-code instruction, micro-code sequencer **135** provides the NLIP, IPdelta, BLIP, branch prediction bit, and MSissue along path **136**. This information is stored by recirculation register **310,** for example, under control of micro-code sequencer **135**. Then, for each of the micro-ops associated with the macro-code instruction, allocation unit **140** provides to multiplexer **315** any information unique to that particular micro-op such as, for example, uip and sequence number. micro-code sequencer **135** then controls multiplexer **315** in such a manner as to provide the information from recirculation register **310**, and the information from allocation unit **140** to TBIT **142**.

Accordingly, in operation, recirculation register **310** may be loaded, for example, only one time for each macro-code instruction processed. If a macro-code instruction decodes

to several micro-ops, the micro-op specific information (e.g., sequence number and uip) is provided to multiplexor **315** as allocation unit **140** assigns each sequence number. Thus, one entry is stored for each micro-op of a macro-code instruction. Moreover, each entry associated with a particular macro-code instruction has, for example, the same information stored in NLIP field **220**, BLIP field **225**, branch prediction bit field **230**, IPdelta field **235** and MSissue **245**. Only the information in the sequence number field **215** and uip field **240** are different.

Alternatively, it is possible for certain groups of instructions to map to the same entry in the table, for example, if the data is the same for each of the instructions

Additionally, as illustrated in Fig. 3, recirculation register **310** is loaded with information from TBIT **142** during both event recovery and micro-branch recovery as is described in further detail below.

Pipeline **100** also includes two additional registers, an event register **320** and a micro-branch register **330**. Each of these registers are coupled to TBIT **142** and to micro-code sequencer **135**. event register **320** and micro-branch register **330** are loaded with information from TBIT **142** during event recovery and micro-branch misprediction recovery, respectively.

Event Recovery: The flowchart of Fig. 4 shows an exemplary process performed in connection with TBIT **142** during event recovery. Assume that the processor events on a particular micro-op. Write back unit **160** detects the event and provides the sequence number of the "current" micro-op (i.e., the micro-op that was next to be retired) to TBIT **142** (step 405). TBIT **142** then loads event register **320** (step 410) and recirculation register **310** (step 415) with information pertinent to the instruction upon which the processor evented. In particular, in the exemplary embodiment, upon the occurrence of an event, TBIT **142** reads the entry **210** associated with the micro-op upon which the processor evented. The appropriate entry **210** is selected by comparing the sequence number of the micro-op that eventED to the sequence numbers in the sequence number field **215**. The selected entry **210** is then loaded into event register **320** (step 410). Simultaneously, the recirculation register **310** is loaded with information from selected fields of the TBIT **142** entry (step 415). For example, the NLIP, BLIP, branch prediction bit, IPdelta and MSISSUE are loaded into recirculation register **310** in this embodiment.

Next, micro-code sequencer **135** provides the appropriate event recovery micro-code to allocation unit **140** for execution (step **420**). In particular, micro-code sequencer **135** provides to allocation unit **140** particular micro-ops associated with event recovery. Allocation unit **140** assigns each micro-op a sequence number, and transmits each sequence number and uip to multiplexer **315**. For each micro-op, micro-code sequencer **135** controls the multiplexer **315** in such a manner as to store the information from recirculation register **310** and the information provided by allocation unit **140** (sequence number and uip) in individual entries in TBIT **142**. This stored information may be needed if the processor events on one of the micro-code micro-ops.

After the event recovery micro-code is executed, micro-code sequencer **135** reads the information stored in the event register **320** and determines which macro-code instruction should be executed once the machine recovers (step 425). In particular, if the event is a fault condition (e.g., a hardware problem is detected), and MSissue= "0" (i.e., the instruction which evented did not originate from micro-code sequencer **135**) the micro-code may determine, for example, that the current macro-instruction should be re-executed. The micro-code then calculates the address of the current instruction from information stored in event register **320**, e.g., NLIP - IPdelta. This address is then transmitted to instruction fetch unit **110**, and the pipeline is restarted.

If the event is a trap condition (e.g., an automatic procedure call initiated by some condition, such as, for example, an overflow condition), and MSissue="0," the micro-code may determine, for example, that the current instruction has completed and that the next instruction should be executed. In particular, if the macro-instruction upon which the machine evented is a branch instruction (the BLIP field has a well defined value), and the branch instruction was predicted as taken (as indicated by the branch prediction bit), the micro-code transmits BLIP (i.e., the branch target address) to instruction fetch unit **110**, for example. If the macro-instruction is not a branch instruction or the macro-instruction is not a branch instruction predicted as taken, NLIP is transmitted to instruction fetch unit **110**.

If, however, MSissue=1, i.e., the instruction originated from micro-code sequencer **135**, the micro-code utilizes the uip to restart.

In any case, once the micro-code makes the determination as to which instruction should be executed next, micro-code sequencer **135** utilizes the information in the event register

320 to determine the appropriate instruction address. The instruction address is then provided to instruction fetch unit 110 (or the decoded cache unit 130) so that instruction fetch unit 110 (or the decoded cache unit 130) can fetch the macro-code instruction.

Micro-Branch Misprediction Recovery: Turning now to the flowchart of Fig. 5, the process performed in connection with TBIT 142 during micro-branch misprediction recovery is illustrated. As described above, for certain micro-branch instructions, micro-code sequencer 135 makes branch direction predictions. Micro-code sequencer 135 then provides allocation unit 140 with micro-ops along only the predicted instruction path. Accordingly, it is possible that micro-code sequencer 135 mispredicted a micro-branch instruction.

Assume that execution unit 150 detects a micro-branch misprediction (step 505). Execution unit 150 provides the sequence number of the mispredicted branch instruction to TBIT 142 (step 505). TBIT 142 then loads micro-branch register 330 (step 510) and recirculation register 310 (step 520) with information pertinent to mispredicted micro-branch instruction. In particular, in the exemplary embodiment, TBIT 142 reads the entry 210 associated with the mispredicted micro-branch instruction. The appropriate entry 210 is selected by comparing the sequence number of the mispredicted micro-branch instruction to each of the sequence numbers in the sequence number field 215. Information from the selected entry 210, such as, for example, the NLIP and IPdelta, is then loaded into micro-branch register 330 (step 510). Simultaneously, the recirculation register 310 is loaded with information from selected fields of the TBIT 142 entry (step 515). For example, the NLIP, BLIP, branch prediction bit, IPdelta and MSISSUE are loaded into recirculation register 310 in this embodiment.

Next, micro-code sequencer 135 provides the appropriate micro-branch misprediction micro-code to allocation unit 140 for execution (step 520). In particular, micro-code sequencer 135 provides to allocation unit 140 particular micro-ops associated with micro-branch misprediction recovery. Allocation unit 140 assigns each micro-op a sequence number, and transmits each sequence number and uip to multiplexer 315. For each micro-op, micro-code sequencer 135 controls the multiplexor 315 in such a manner as to store the information from recirculation register 310 and the information provided by allocation unit 140 (sequence number and uip) in individual entries in TBIT 142. This stored information may be needed if the processor events on one of the micro-code micro-ops.

Finally, after the branch misprediction recovery micro-code is executed, micro-code sequencer **135** reads micro-branch register **330** and determines which macro-code instruction should be executed once the machine recovers based on, for example, the information stored in micro-branch register **330** (step 525).